基于 IFOA-GA 任务调度算法在云计算 MapReduce 模型中的研究 *

陈　暄[1]，潘春平[1]，龙　丹[2]

(1. 浙江工业职业技术学院，浙江 绍兴 312000; 2. 浙江大学，杭州 310058)

摘　要：针对传统的云计算任务调度算法存在效率低，利用率不高的问题，采用改进的果蝇算法(improved fruit fly optimization algorithm，IFOA)和遗传算法(genetic algorithm，GA)融合的算法用于处理任务调度。首先，将任务调度转换为 DAG(directed acyclic graph，DAG)并通过 Kruskal 算法将任务调度顺序进行化简；其次，针对果蝇算法的种群采用正交数组和量化技术进行初始化，对果蝇算法边界进行处理，对探索步长进行动态调整，并使用 GA 算法对个体选择进行选择处理；最后，将融合后生成的算法 IFOA-GA 用于仿真平台中的云计算任务调度，相对于 IGA，IFOA，IPSO 算法在 QoS 的四个指标对比中具有一定的优势，说明 IFOA-GA 算法能够有效的提高云计算调度效率。

关键词：云计算；任务调度；果蝇算法；种群初始化；边界处理

中图分类号：TP301　　doi: 10.3969/j.issn.1001-3695.2018.06.0307

Research on cloud computing MapReduce model based on IFOA-GA task scheduling algorithm

Chen Xuan[1], Pan Chunpin[1], Long Dan[2]

(1. *Zhejiang Industry Polytechnic College*, *Shaoxing Zhejiang* 312000, *China*; 2. *Zhejiang University*, *Hangzhou* 310058, *China*)

Abstract: Aiming at the low efficiency and low utilization rate of traditional cloud computing task scheduling algorithms, this paper proposed an improved algorithm using the improved fruit fly optimization algorithm (IFOA) and genetic algorithm (GA) for task scheduling. Firstly, the task scheduling is converted into a DAG (Directed Acyclic Graph) , and the task scheduling sequence is simplified through the kruskal algorithm. Secondly, the population of Drosophila algorithm is initialized using orthogonal arrays and quantization techniques. The boundaries of the Drosophila algorithm algorithm are processed, the exploration step size is dynamically adjusted, and the individual selection is processed using the GA algorithm. Finally, the fusion algorithm IFOA-GA is used in cloud computing task scheduling in the simulation platform. Compared with IGA, IFOA and IPSO algorithm, it has certain advantages in the comparison of four indexes of QoS, which shows that the IFOA-GA algorithm can be effective in mproving cloud computing scheduling efficiency

Key words: cloud computing; task scheduling;Drosophila algorithm;population initialization;boundary processing

0　引言

云计算任务调度一直以来都是云计算中研究的重要组成部分,其本质问题就是一种多目标优化问题,也是一种 NP 问题,传统的任务调度方法显然无法适应云计算任务调度,因此采用智能算法来求解是目前研究的主要方向[1]。国内外学者针对云计算任务调度算法进行了研究,在蝙蝠算法[2],遗传算法[3],蚁群算法[4-5],粒子群算法[6]中都对其进行了改进,取得了不错的效果;文献[7]提出了一种基于烟花算法的多目标优化调度模型,仿真实验说明了该算法具有良好的调度效率,该算法优点是利用了烟花算法性能高的特点,合理而快速的进行调度,缺点是仅仅与基本 PSO,GA 算法相比,缺乏对同类型的最新的算法进行对比;文献[8]提出了基于蝙蝠算法的云计算任务调度方法,通过对蝙蝠算法的种群初始化和依靠 Powell 局部搜索提高了云计算下的调度性能,该算法的优点是提高了虚拟机的处理效率,缺点是仅仅与聚类算法进行对比,缺乏与其他算法对比效果,文章说服力不够;文献[9]提出在云计算中采用改进的遗传算法, 通过双向收敛蚁群算子求出精确解, 该算法优点提高了求解精度和收敛速度, 缺点是增加了算法的复杂性, 提高了算法的求解时间；文献[10]提出将蚁群算法和粒子群算法进行改进后融合, 用于云计算资源调度效率, 该算法的优点是降低了消耗时间, 降低了成本, 缺点是降低了算法的精度。

收稿日期：2018-06-04；**修回日期：**2018-07-17　**基金项目：**国家自然科学基金资助项目（LQ18A010003，11426205）；绍兴市科技局项目（2015B70013）
作者简介：陈暄（1979-），男，江西人，副教授，硕士，主要研究方向为云计算、无线传感；潘春平（1978.3-），男，湖南人，副教授，硕士，主要研究方向为算法设计；龙丹（1975-）男，湖南人，讲师，博士，主要研究方向为图像处理和分析、算法设计.

本文在以上研究成果的基础上,提出了将改进的果蝇算法用于云计算的任务调度中。仿真实验通过与其他智能算法对比,取得了较好的效果。

1　云计算中的任务调度

1.1　基于 MapReduce 模型的任务定义

目前,大部分云计算任务模型都是采用基于 MapReduce 的思想,它是一种高效的任务调度模型,具体的工作流程已经在很多关于 MapReduce 文献中得到了体现,这里就不再进行描述了,在 MapReduce 模型中,具有一定不足[11],其表现在:(1)用户提交的任务并不是都是简单的任务,(2)用户提交的任务不一定都能够分解成并行的子任务,子任务之间存在偏序的关系;(3) 在 Reduce 中存在只有内层迭代完成才能进行外层迭代的问题。针对这些问题,本文对于 MapReduce 模型进行了一些新的定义

定义 1:定义复杂任务　所谓复杂任务是一组存在依赖关系的子任务构成的系统,该任务系统定义为 $(T,\prec,D,A)$

(1) $T=(t_1,t_2,\cdots t_n)$ 是一组可执行的任务; t_i 是任务 T 中子任务

(2) $\prec$ 是 T 上一个偏序关系,主要用来说明子任务之间的优先级,当 $t_i \prec t_j$,说明 t_i 要在 t_j 之前开始执行

(3) D 为通信矩阵, $d_{ij}>=0$ 表示从 t_i 到 t_j 的数据量

(4) A 是 n 维向量, $A_i>0$ 表示任务 t_i 的计算量

显然云计算中作为组成复杂任务的子任务之间存在优先约束关系,复杂任务可以采用 DAG 图来表示, $G=(T,E)$, T 表示图中节点的集合,节点权值表示任务处理时间, E 为边集合,边权值表示了数据之间依赖关系和通信时间,因此,一个 DAG 优先约束关系是一个节点在执行前,它的前序节点必须已经执行,当两个节点分配到同一个处理器,则他们的通信成本为 0。

定义 2:当 G 为有向图,把以顶点 v 为终点的边的数目,记作 $ID(v)$,把以顶点 v 为起点边的数目,记作 $OD(v)$;

定义 3:在一个 DAG 图的生成树是必须满足如下条件的子图 $T:G$ 和 T 具有相同的顶点数,在 T 中有足够的边连接 G 中的所有顶点,当没有回路。当 DAG 图的每条边都指定一个权,因此集合中所有边的权最小生成树就是最小生成树。本文采用 kruskal 算法将 DAG 图转换为最小生成树,其处理过程如下:图 1(a)中表示 DAG 图中描述的任务关系,图 1(b)描述了最小生成树。两个图中的节点完全相同,有效的将 DAG 图中的冗余依赖关系精简了。

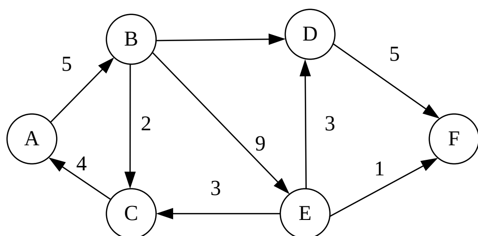


图 1　(a) DAG 之间的任务关系

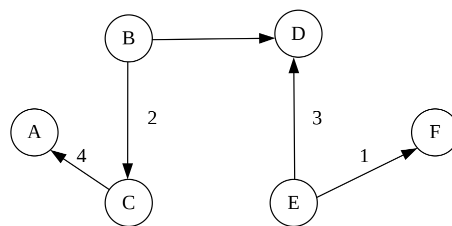


图 1　(b)生成最小生成树

1.2　构建 QOS 的任务调度评价模型

本文选择基于服务质量 QoS(Qualtiy of Service)作为 MapReduce 模型中的任务调度的效果一个重要机制,在一般的 QoS 的基础上,将完成时间、花费成本,可靠性和能量消耗作为任务评价指标。

(1)完成时间:所谓完成时间主要是指一个任务在整个云计算资源调度中所花费的时间,一般而言,云计算的完成时间是用户最关心的问题,是衡量云计算任务调度效果好坏的重要标准。

(2)花费成本:云计算的任务调度肯定需要消耗云计算中若干资源,诸如 CPU,内存,硬盘等等,这些资源具有各自的使用成本。因此云计算中的任务尽可能的在最低限度的消耗成本来保证任务的完成。

(3)可靠性:由于云计算自身就是一个将出于不同物理位置的计算机变成逻辑位置在一起的虚拟计算机系统,因此整个系统的可靠性是云计算任务调度的基础,由于系统可靠性无法采用具体的公式进行衡量,因此通过用户满意度来反映系统可靠性;

(4)能量消耗:云计算任务调度过程中不可避免的会涉及到能量消耗,这是因为云中不同的任务在调度不同的资源过程中,伴随着网络的带宽、硬件方面的消耗产生了能量消耗,在任务调度中,能量消耗是一个不能被忽视的问题。

为了能够获得高效的 QoS,必须在同时满足以上四个评价标准的基础上,设 T,C,S,E 分别表示以上指标评价的完成时间、花费成本、可靠性和能量消耗,并且 $r1,r2,r3,r4$ 代表其权重,且 $r1+r2+r3+r4=1$,使用 $F(i)$ 表示第 i 个任务调度方案,如公式(1)所示,当找到所有任务中的最优函数值即 F_{best} 才是基于 QoS 的最佳的任务调度方案,如公式(2)所示。

$$F(i)=r1\bullet T_i+r_2\bullet C_i+r_3\bullet S_i+r_4\bullet E_i \tag{1}$$

$$F_{best}=Best(F(1),F(2)\cdots F(i)) \tag{2}$$

2　FOA 算法

2011 年,Pan 提出了一种新型的群体智能优化算法:果蝇优化算法(Fruit Fly optimization algorithm,简称 FOA),该算法主要模拟的是果蝇种群的觅食行为,采用基于群体协作机制进行寻优操作。FOA 算法大致分为如下几个步骤

步骤 1:设置种群规模 $Popsize$,最大迭次数 Num ,群体范围 Lr 和果蝇个体单次飞行范围 Fr ,果蝇群体中的个体的位置信息有对应的二维坐标给出,初始位置如下:

$$\begin{cases} X_axis = rand(Lr) \\ Y_axis = rand(Lr) \end{cases} \tag{3}$$

步骤 2:嗅觉搜索过程

步骤 2.1 当群体中的一只果蝇通过嗅觉进行搜索的时候,赋值一个随机的飞行方向和距离,因此果蝇个体 i 的新位置为:

$$\begin{cases} X_i = X_axis + rand(Fr) \\ Y_i = T_axis + rand(Fr) \end{cases} \tag{4}$$

步骤 2.2 食物的味道距离果蝇个体的位置是位置的,因此先计算果蝇个体到原点的距离 $Dist_i$,如公式(5),然后通过公式(6)设定味道的浓度值 S_i

$$Dist_i = \sqrt{X_i^2 + Y_i^2} \tag{5}$$

$$S_i = 1/Dist_i \tag{6}$$

步骤 2.3:通过公式(7)计算当前群体中的每一个果蝇个体 i 的味道浓度 $Smell_i$,

$$Smell_i = fitness(S_i) \tag{7}$$

$fitness(S_i)$ 表示浓度判断函数,通过 FFOA 算法优化问题求解,它是目标函数。

步骤 2.4: 选择当前种群中的最佳浓度值的果蝇个体,记录其味道的浓度值和相应位置

$$[bestSmell, bestIndex] = \min(Smell) \tag{8}$$

步骤 3:视觉搜索过程

果蝇群体飞向步骤 2 中的最佳浓度的相应位置,即

$$SmellBest = bestSmell \tag{9}$$

$$\begin{cases} X_axis = X(bestIndex) \\ Y_axis = Y(bestIndex) \end{cases} \tag{10}$$

步骤 4:不断的重复执行步骤 2 和步骤 3,直到算法达到最大迭代次数。得到的最佳浓度的位置就是 FFOA 算法的最优解。

FOA 算法的优点是整体算法简单,主要包括嗅觉搜索和视觉搜索两个部分,重要的算法参数仅为种群的数目和最大迭代搜索次数,文献[12-13]说明了 FOA 算法相比于其他智能算法具有良好的全局搜索策略和优化能力,但算法存在的缺点是局部搜索能力不足,寻优结果不稳定导致算法过早收敛。

3　基于 IFOA-GA 算法的云计算任务调度策略

针对 FOA 算法存在的问题,本文从种群初始化、边界处理、步长变换和个体选择等 4 个方面进行改进,同时引入了 GA 与之进行融合,生成了新的算法 IFOA-GA 算法,并将此算法用于云计算任务调度中。

3.1　种群初始化

实验表面对种群进行初始化能够对最优解具有积极的推动作用,能够有效的避免算法在前期中产生很多的无效的解[14],显然,FOA 是没有进行种群初始化的,为了能够有效的避免这种情况的发生,保证有效解能够均匀额分布在解的空间中,特次需要对种群进行初始化。本文采用基于正交数组和量化技术的正交实验方法来进行 FOA 种群的初始化,这种方法可以使得初始解能够均匀的分布在可行解中,提高算法搜索最优解的速度。

步骤 1: 将 FOA 种群问题可行域 $[l,u]$ 划分为 $[l_1,u_1],[l_2,u_2],\cdots[l_S,u_S]$,具体划分为

$$\begin{cases} l_i = l + (i-1)\left(\dfrac{u(s)-l(s)}{S}\right) l_S \\ u_i = u - (S-i)\left(\dfrac{u(s)-l(s)}{S}\right) l_S \end{cases}, i = 1,2\cdots S \tag{11}$$

其中, $u(s)-l(s) = \max\limits_{1\le i\le D}\{u_i - l_i\}$

步骤 2:将子集 $[l_i,u_i]$ 进行 Q_1 次量化后,得到

$$a_{ij} = \begin{cases} l_i, & j=1 \\ l_i + (j-1)\left(\dfrac{u_i - l_i}{Q_1 - 1}\right) & 2 \le j \le Q_1 - 1 \\ u_i & j = Q_1 \end{cases} \tag{12}$$

其中, Q_1 是奇数,构造正交数组 $L_{M_1}(Q_1^N) = \left[a_{ij}\right]_{M_1\times N}$,根据公式(13)

$$\begin{cases} (a_{1,a_{11}}, a_{2,a_{12}}, \cdots, a_{N,a_{1N}}) \\ (a_{1,a_{21}}, a_{2,a_{22}}, \cdots, a_{N,a_{2N}}) \\ \cdots \\ (a_{1,a_{M_1 1}}, a_{2,a_{M_1 2}}, \cdots, a_{N,a_{M1N}}) \end{cases} \tag{13}$$

选择 M_1 个个体,其中 $L_{M_1}(Q_1^N)$ 按照如下规则进行构造,当满足 $(Q_1^{J_1}-1)/(Q_1-1) \ge N$ 的最小 J_1 。当 $(Q_1^{J_1}-1)/(Q_1-1) = N$,则 $N^{'} = N$,否则, $N^{'} = (Q_1^{J_1}-1)/(Q_1-1)$,构造数组基本列 $j = \dfrac{Q_1^{k-1}-1}{Q_1-1}+1$, $a_{ij} = \left\lfloor \dfrac{i-1}{Q_1^{J_1-k}} \right\rfloor \bmod Q_1$, $i=1,\cdots,M_1, k=1,\cdots,J_1$,非基本列为 $j = \dfrac{Q_1^{k-1}-1}{Q_1-1}+1$, $a_{j+(s-1)(Q_1-1)+t} = (a_s \times t + a_j) \bmod Q_1$, $s=1,\cdots,j-1, t=1,\cdots,Q_1$,这样数组 $L_{M_1}(Q_1^{N^{'}})$ 就构造完成。删除 $L_{M_1}(Q_1^{N^{'}})$ 的最后 $N^{'}-N$ 列得到 $L_{M_1}(Q_1^N)$,其中, $M_1 = Q_1^{J_1}$

步骤 3:在 $M_1 S$ 个体中,选择 SN 个适应度最高的个体作为初始种群。

3.2　边界处理

FOA 中个体极其有可能会超出可行域的范围的边界,当某个个体 x_i 的第 k 维超出边界时,则按照公式(4)的处理方法将超过边界的果蝇个体映射到一个新的位置,定义如下:

$$\hat{x}_{i,k}=\begin{cases}x_{o,k}+\dfrac{x_{i,k}-x_{o,k}}{\|x_i-x_o\|}*|x_{LB,k}-x_{o,k}| & if\ x_{i,k}<x_{LB,k}\\ x_{o,k}+\dfrac{x_{i,k}-x_{o,k}}{\|x_i-x_o\|}*|x_{UB,k}-x_{o,k}| & if\ x_{i,k}>x_{UB,k}\end{cases}\tag{14}$$

式(14)中，$x_{UB,k},x_{LB,k}$ 分别为解空间中的第 k 维的上下边界，x_o 表示解空间的原点。公式(14)借助图 2 的推理的过程如下:

(1)越上界

$$\cos\alpha=\frac{x_{i,k}-x_{o,k}}{\|x_i-x_o\|}=\frac{m}{|x_{UB,k}-x_{o,k}|}$$
$$\Rightarrow m=\frac{x_{i,k}-x_{o,k}}{\|x_i-x_o\|}*|x_{UB,k}-x_{o,k}|\tag{15}$$

以原点 x_o 作为中心,半径为 m 进行画圆,通过与 k 轴的交点就是超越边界处理后的第 k 维的分量,即

$$\hat{x}_{i,k}-x_{o,k}=m=\frac{x_{i,k}-x_{o,k}}{\|x_i-x_o\|}*|x_{UB,k}-x_{o,k}|$$
$$\Rightarrow \hat{x}_{i,k}=\frac{x_{i,k}-x_{o,k}}{\|x_i-x_o\|}*|x_{UB,k}-x_{o,k}|+x_{o,k}\tag{16}$$

(2)超越下边界

$$\cos\beta=\frac{x_{i,k}-x_{o,k}}{\|x_i-x_o\|}=\frac{n}{|x_{LB,k}-x_{o,k}|}$$
$$\Rightarrow n=\frac{x_{i,k}-x_{o,k}}{\|x_i-x_o\|}*|x_{LB,k}-x_{o,k}|\tag{17}$$

与超越上边界一样,它以原点 x_o 作为中心,半径为 n 进行画圆,通过与 k 轴的交点就是超越边界处理后的第 k 维的分量,即

$$\hat{x}_{i,k}-x_{o,k}=n=\frac{x_{i,k}-x_{o,k}}{\|x_i-x_o\|}*|x_{LB,k}-x_{o,k}|$$
$$\Rightarrow \hat{x}_{i,k}=\frac{x_{i,k}-x_{o,k}}{\|x_i-x_o\|}*|x_{LB,k}-x_{o,k}|+x_{o,k}\tag{18}$$

通过上面的推理可以发现,超越上边界的果蝇个体进行处理处理后,其位置能够能够靠近上边界,同理,当超越下边界的果蝇个体处理后,其位置就会靠近下边界,这样相对位置没有发生改变,保持了一些原有的数据特性。图 2 表示了越界处理坐标参考,图 3 表示假设一定范围后的越界前后的对比效果。

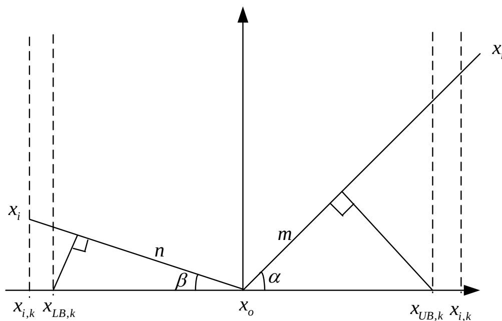


图 2 越界处理辅助示意图

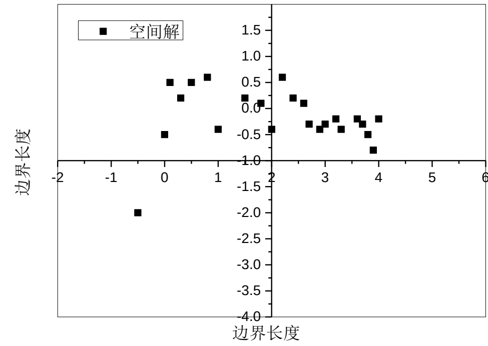


图 3 (1) 越界前示意图

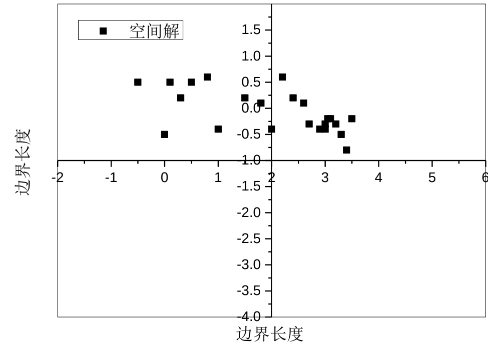


图 3 (2) 越界后示意图

3.3 搜索步长的改进

果蝇算法[15]会在果蝇新位置的诞生中设定一定的步长,这样设置的好处在于能够使得果蝇个体在向随机方向搜索的时候,不会超出一定的范围。当步长设置的过大的时候,算法的全局搜索能力就会较强,从而收敛速度快,但局部搜索能力降低,导致算法在后期收敛精度不够,当设置步长过小的时候,局部搜索能力过大,但全局搜索能力降低。因此,本文对步长采用可变换的思想进行改进,如公式(19)所示。

$$\begin{cases}X_i=X_axis+rand(Fr)\times H\\ Y_i=T_axis+rand(Fr)\times H\end{cases}$$
$$s.t\ \ H_i=\begin{cases}L\times(1+\sin(i))^{\alpha} & H_i\le L\\ L & H_i>L\end{cases}\tag{19}$$
$$\alpha=\mathrm{mod}(i,T)$$

式中，L 为算法搜索空间的长度，T 为搜索迭代次数，α 为第 i 次迭代取余，因此 H_i 分布如图 4 所示

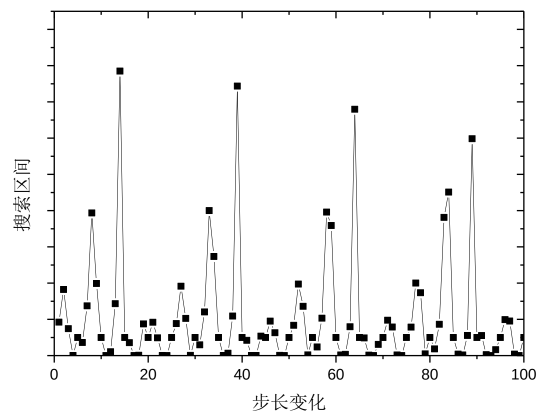


图 4 步长变化

从图 4 中发现,改进的步长方式将整个搜索过程分解为若干个周期,这样能够有效的增强搜索过程中的多样性,使得算法能

够跳出局部收敛,减少了局部收敛的可能性,采用 $\sin(x)$ 函数,使得步长在单位周期内进行有规律的变化,能够有效的控制步长在增加和减少对算法带来的影响,当 $\sin(x)$ 在单调递增的时候,步长逐渐增大,使得算法具有较强的全局搜索能力,能够解决 $\sin(x)$ 在上一个单调递减内可能存在局部收敛的问题,在 $\sin(x)$ 函数递减区,步长逐渐减少,使得算法能够在小范围内完成高精度的搜索,具有更好的收敛效果。

3.4　个体选择改进

在 FOA 算法中,果蝇个体仅仅是凭借自身的浓度来选取方向,显然这种方式仅仅是从果蝇个体的角度出发,没有考虑到果蝇个体自身的优劣为在下一代迭代中产生的影响,没有从整体上考虑最优浓度值对整个算法的影响。因此,借助 GA 思想对果蝇个体选择进行改进。

(1)果蝇个体选择

果蝇个体根据浓度值来确定适应度值,并且将个体的最优的浓度值作为当前最优适应度,这样的方式存在一定的缺陷,主要是没有考虑到果蝇个体的浓度与整个种群对应的浓度的关系,在一定程度上造成算法解陷入布局最优。本文采用基于正向序列的适应度函数的构造,按照目标函数值的大小将种群中的个体按照从大到小的顺序排序,然后将排序后的个体按照映射关系计算适应度函数值。构造正向序列的适应度函数为

$$f_i = 1-\frac{i+1}{N}, i=1,2,\cdots N \tag{20}$$

式(20)中 i 表示果蝇种群中的个体, N 为种群规模,两个相邻的个体之间的适应度值差为:

$$f_{i+1}-f_i = -\frac{1}{N} \tag{21}$$

式(21)表明相邻个体之间的差异取决于种群的规模,因此按照轮盘选择个体方式来计算概率,第 i 个个体被选择的概率为:

$$p_i = \frac{N-i+1}{\sum_{i=1}^{N} N-i+1}, i=1,2,\cdots N \tag{22}$$

两个相邻的个体被选择的概率误差为:

$$p_{i+1}-p_i = -\frac{1}{\sum_{i=1}^{N} N-i+1}, i=1,2,\cdots N \tag{23}$$

$$f(x_i) = \sum_{i=1}^{n} p_i * (y_i^{'} - y_i) \tag{24}$$

从式(24)中可以发现,相邻个体之间被选择的概率比较小,这就保证不会因为个体之间被选择的概率过大而影响选择操作,特别是当种群规模增大的时候,差异就越小,从而提高个体选择的能力,从而能够提高个体被选择的几率,增加了多种的多样性。

(2)变异操作:将第 g 代种群中的第 i 个果蝇个体 x_i^g 依据公式(25)的变异方式。

$$V_i^{g+1} = x_i^g + F*(x_{r1}^g - x_{r2}^g) \tag{25}$$

式(25)中的 V_i^{g+1} 是变异后的种群中的个体, F 为随机因子用来控制缩放程度,设定值为[0,1]之间。

(3)交叉操作:通过一定的概率选择,将变异的中第 i 个体 x_i^g 与父代个体 V_i^{g+1} 之间在第进行交叉,得到新的个体

$$\kappa_i^g = \begin{cases} V_i^{g+1}, & t\in[0,1] \\ x_i^g, & otherwise \end{cases} \tag{26}$$

式(26)中可以保证在交叉过程出现一个 0 到 1 之间的随机整数,能够保证 $\kappa_{i,j}$ 至少有一个分量来自 $y_{i,j}$ 。

3.5　算法流程

步骤 1:将云计算任务调度按照 DAG 图描述,根据 kruskal 算法精简为最小生成树,去掉冗余的信息,保留最精简的任务关系,并根据 QoS 模型设定好每一个任务的函数值,并与果蝇个体一一对应,当寻找到最佳果蝇个体即为最优的任务调度函数值,初始化果蝇算法中的相关参数

步骤 2:采用公式(11-13)对果蝇种群进行初始化

步骤 3:采用公式(19)对果蝇嗅觉搜索过程进行位置更新,执行公式(5-8),在视觉搜索过程中,如果飞行过程中出现越界情况,按照公式(14)的越界处理。

步骤 4:对果蝇个体采用(20-26)进行更新

步骤 5:判断算法是否达到最大迭代次数,如果达到了,就转步骤 6,否则转步骤 3

步骤 6:算法结束,最优果蝇个体即为最佳的任务函数值。

流程如图 5 所示:

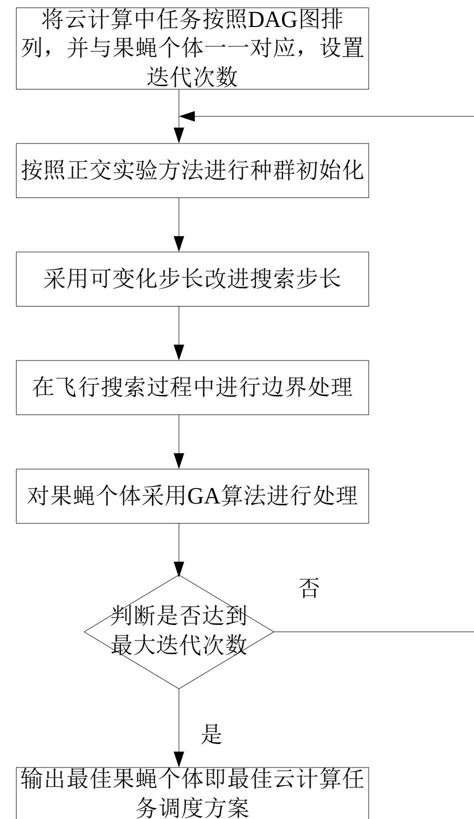


图 5　本文算法流程图

3.6　算法仿真

为了进一步说明 IFOA-GA 算法在云计算中具有的效果,本

文选择硬件平台中的 CPU 为酷睿 i57500,内存为 4GDDR3,硬盘为 1T,软件环境为 64 位的 Windows 系统,仿真平台采用 Cloudsim。将 IFOA-GA,IFOA[15],IPSO[16]和 IGA[17]算法进行比较,设置算法的最大迭代次数为 100 次,任务数量为[10000,100000]。参与对比的算法的参数均以各自的文献为主。从完成时间、花费成本、可靠性和能量消耗四个方面进行对比。

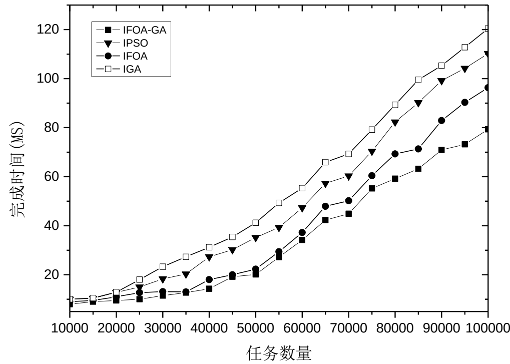


图 6　四种算法的完成时间对比

图6显示了四种算法的完成时间对比,伴随着任务数量的逐渐增多,四种算法的完成时间都在逐渐增多, IPSO,IGA 都是改进后的智能算法,但是与 IFOA-GA 相比还是存在的一定的差距,从整个任务完成的情况来看,IFOA-GA 所需要完成的时间还是最少的,这说明 IFOA-GA 算法相比其他三种算法具有一定的优势。

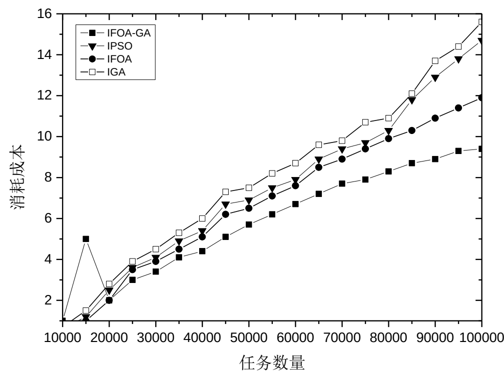


图 7　四种算法的消耗成本对比

图7显示了四种算法的消耗成本对比,随着任务数量的逐渐增多,四种算法消耗成本都在逐渐的增多,IFOA-GA 算法在初始阶段消耗的成本的曲线波动比较大,究其原因是因为算法在开始阶段进行种群的初始化,消耗了较多的成本,后期随着算法逐渐深入,算法消耗成本逐步稳定,相比其他三种算法,IFOA-GA 算法在总体上还是处于一定的优势,这也说明 IFOA-GA 算法相比于其他三种算法能够有效的降低成本。

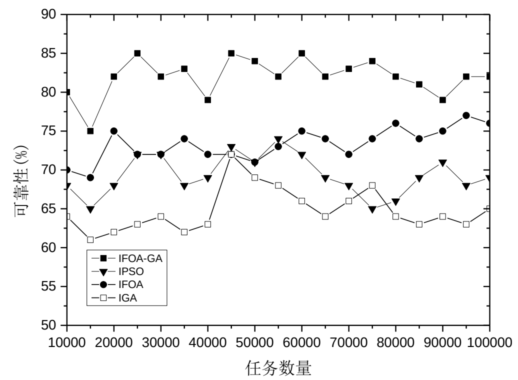


图 8　四种算法可靠性

图8显示了四种算法的可靠性的对比结果,随着任务数量的逐渐增多,IFOA-GA 算法相比于其他三种算法具有一定的优势,这是因为 IFOA-GA 算法对 FOA 算法进行了改进,并结合 GA 优点,因此算法的性能更加突出,在云计算的任务调度中具有良好的效果。

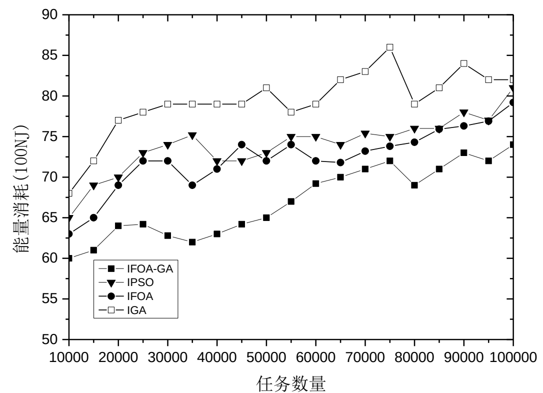


图 9　四种算法的能量消耗对比

图9显示了四种算法的能量消耗的对比结果,随着任务数量的逐渐增多,四种算法的能量消耗都在逐渐的增大,但是从整体上看,IFOA-GA 算法能量消耗曲线稳定,波动性小,而其他三种算法的曲线波动性大,因此,IFOA-GA 算法相比其他三种算法大致平均降低了 7.27%,9.33%和 15.49%。

在以上的实验结果中说明了本文算法相比于 IFOA,IPSO 和 IGA 算法具有一定的优越性,通过在果蝇的种群初始化中使用正交数组和量化技术提高了算法最优解的速度,对果蝇个体可能出现的越界进行处理,对果蝇探索步长进行动态调整,使用 GA 算法对个体选择进行处理等措施,使得算法的整体性能得到提升,也进一步提高了云计算下的任务调度效率。

4　结束语

针对云计算任务调度存在的问题,本文在 FOA 和 GA 基础上进行算法融合生成了 IFOA-GA 算法,将该算法运用在云计算任务调度中获得较好的效果,但本文没有考虑虚拟机负载在任务调度中的影响,在下一步中将继续展开研究。

参考文献:

[1] 张建勋, 古志民, 郑超. 云计算研究进展综述 [J]. 计算机应用研究,

2010, 27 (2): 429-433. (Zhang Jianxun. Survey of research progress on cloud computing [J] , Application Research of Computers, 2010, 27 (2): 429-433.)

[2] Santwana S, Saurabh B. Workflow Scheduling in Cloud Computing Environment Using Bat Algorithm [J] , Proceedings of First International Conference on Smart System, Innovations and Computing, 2018, Doi. org//10. 1007//978-981-10-5828-8_15

[3] Kaur K. A Novel Context and Load-Aware Family Genetic Algorithm Based Task Scheduling in Cloud Computing [J] , Data Engineering and Intelligent Computing, 2017, Doi. org//10. 1007//978-981-10-3223-3_51

[4] Vinothina V. An Approach for Workflow Scheduling in Cloud Using ACO [J] , Big Data Analytics, 2017, Doi. org//10. 1007//978-981-10-6620-7_50

[5] Shabeera T P. Optimizing VM allocation and data placement for data-intensive applications in cloud using ACO metaheuristic algorithm [J] , Engineering Science and Technology, an International Journal, 2017, 20 (2): 616-628

[6] Sadhasivam N. Cancer Diagnosis Epigenomics Scientific Workflow Scheduling in the Cloud Computing Environment Using an Improved PSO Algorithm [J] Asian Pac J Cancer Prev. 2018; 19 (1): 243–246.

[7] 黄伟建, 郭芳. 基于烟花算法的云计算多目标任务调度 [J]. 计算机应用研究, 2017, 34 (6): 1718-1720. (Huang Jinwei, Guo Fang. Multi-objective task scheduling based on fireworks algorithm in cloud computing [J] , Application Research of Computers, 2017, 34 (6): 1718-1720.)

[8] 乔良, 林伟伟. 基于改进蝙蝠算法的云计算任务调度研究 [J]. 微电子学与计算机. 2017, 34 (7): 27-32. (Qiao Liang, Lin wei-wei. Cloud Computing Task Scheduling Based on Improved Bat Algorithm [J] , Microelectronics & Computer, 2017, 34 (7): 27-32)

[9] 赵俊普, 殷进勇, 金同标. 遗传蚁群算法在云计算资源调度中的应用 [J]. 计算机工程与设计, 2017, 38 (3): 693-697. (Zhao jun-pu, yin jin-yong, Jin tong-biao. Application of genetic ant colony algorithm in cloud computing resource scheduling [J] , Computer Engineering and Design, 2017, 38 (3): 693-697.)

[10] 萨日娜. 基于蚁群粒子群优化算法的云计算资源调度方案 [J], 吉林大学学报: 理学版, 2017, 55 (6): 1518-1522. (Sa Rina, Cloud Computing Resource Scheduling Scheme Based on Ant Colony Particle Swarm Optimization Algorithm [J] , Journal of Jilin University: Sci Ed, 2017, 55 (6): 1518-1522.)

[11] 方锦明. 一种面向云计算的改进的 Mapreduce 模型 [J]. 计算机测量与控制, 2012, 20 (5): 1417-1419. (Fang jing-ming. An improved Mapreduce Models Based on Cloud Computing [J] , Computer Measurement & Control, 2012, 20 (5): 1417-1419.)

[12] 吴小文, 李擎. 果蝇算法和 5 种群智能算法的寻优性能研究 [J]. 火力与指挥控制, 2013, 38 (4): 17-20. (Wu xiao-wen, Li qing. Research of Optimizing Performance of Fruit Fly Optimization Algorithm and Five Kinds of Intelligent Algorithm [J] , Fire Control & Command Control, 2013, 38 (4): 17-20.)

[13] 刘立群, 韩俊英, 代永强. 果蝇优化算法优化性能对比研究. 计算机技术与发展, 2015, 25 (8): 94-98. (Liu li-qun, Han jun-ying, Dai yong-qiang. Comparative Study on Optimization Performance of Fruit Fly Optimization Algorithm [J] , Computer Technology and Development, 2015, 25 (8): 94-98.)

[14] 常天庆, 白帆, 李勇. 解 WTA 问题群智能优化算法的种群初始化问题研究 [J]. 计算机应用研究, 2013, 30 (5): 1377-1380. (Chang tian-qing, Bai fang, Li yong. Research on population initialization for swarm intelligence optimization solving WTA [J] , Application Research of Computers, 2013, 30 (5): 1377-1380.)

[15] 段艳明, 肖辉辉. 一种新的自适应步长果蝇优化算法 [J]. 河南师范大学学报 (自然版) , 2016, 44 (1): 161-168. (Duan yan-ming, Xiao hui-hui. Research of the Self-adaptive Step Fruit Fly Optimization Algorithm [J] , Journal of Henan Normal University (Natural Science) , 2016, 44 (1): 161-168.)

[16] 王晓天, 韩啸. 基于改进粒子群算法的云计算服务部署优化 [J]. 吉林大学学报 (理学版) , 2017, 55 (2): 393-397. (Wang xiao-tian, Han xiao. Cloud Computing Service Deployment Optimization Based on Improved Particle Swarm Algorithm [J] , Journal of Jilin University: Sci Ed, 2017, 55 (2): 393-397.)

[17] 李超, 戴炳荣, 旷志光. 云计算环境下基于改进遗传算法的多维约束任务调度研究 [J]. 小型微型计算机系统, 2017, 38 (9): 1945-1949. (Li chao, Dai bing-rong. Research on Task Scheduling with Multiple Constraints Based on Genetic Algorithm in Cloud Computing Environment [J] , Mini-micro Systems, 2017, 38 (9): 1945-1949.)